Computer Science Department

TECHNICAL REPORT

Mapping Nested Loop Algorithms into Multi-dimensional Systolic Arrays

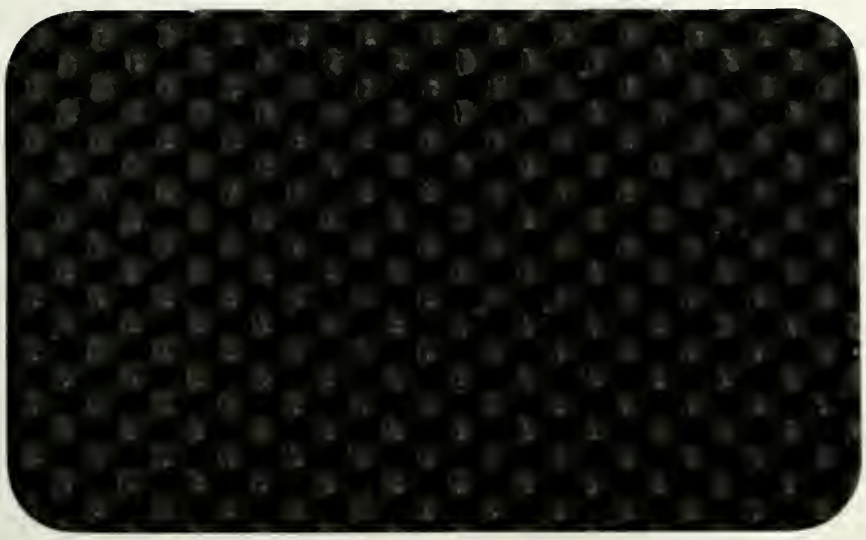
P. Lee
Z. Kedem

Technical Report 395

September 1988

NEW YORK UNIVERSITY

Department of Computer Science
Courant Institute of Mathematical Sciences
251 MERCER STREET, NEW YORK, N.Y. 10012



Mapping Nested Loop Algorithms into Multi-dimensional Systolic Arrays

P. Lee
Z. Kedem

Technical Report 395

September 1988

This work was partially supported by the Office of Naval Research under contract N00014-85-K-0046.
Authors' electronic addresses: leepe@csd2.nyu.edu and kedem@nyu.nyu.edu

Mapping Nested Loop Algorithms into Multi-dimensional Systolic Arrays*

PeiZong Lee and Zvi M. Kedem†

Computer Science Department
Courant Institute of Mathematical Sciences
New York University
251 Mercer Street
New York, NY 10012-1185

September 10, 1988

Abstract

This paper is concerned with transforming depth p nested for loop algorithms into special purpose q-D systolic VLSI arrays, where $1 \leq q \leq p-1$. Previously, techniques have been developed for transforming a p nested loop algorithm into a $(p-1)$-D systolic array. The authors have also developed a technique for transforming a p nested loop algorithm into a 1-D (linear) systolic array. However, the problem of transforming a p nested loop algorithm into a q-D systolic array, where $1 < q < p-1$, has not been studied systematically before. In this paper, we close this gap by giving necessary and sufficient conditions for transforming a p nested loop algorithm into a q-D systolic array in the general case for any q, $1 \leq q \leq p-1$.

Index Terms— algorithm transformations, data contention, data dependence, systolic arrays, parallel processing, VLSI.

*This work was partially supported by the Office of Naval Research under the contract N00014-85-K-0046.
†Authors' electronic addresses: leepe@csd2.nyu.edu and kedem@nyu.nyu.edu

1 Introduction

This report is concerned with solving an open problem of transforming a p nested loop algorithm into a q-D systolic array in the general case, where $1 \leq q \leq p-1$. Transforming loops into systolic (array) algorithms is quite an interesting and challenging problem, which has been extensively studied [1] [2] [4] [8] [9] [10] [11] [12] [13] [14] [15] [16]. The fundamental technique used in these papers is the hyperplane method [3] [5]. They found the data-dependence vectors [3] [5] of an algorithm first, and then they found a nonsingular linear mapping preserving the data-dependence ordering of the original algorithm. This method is also called the space-time mapping.

A depth p nested loop algorithm can be naturally seen as an algorithm to solve a problem with a p dimensional problem space. [2] mapped a p dimensional problem space to a 1-D time hyperplane and an q-D space hyperplane (systolic algorithm) mapping, where $1 \leq q \leq p-1$. [4] [10] [11] mapped a p dimensional problem space to a t loops 1-D time hyperplane and an s-D space hyperplane mapping, where $p = t + s$. [16] mapped a p dimensional problem space to a $p-2$ loops 1-D time hyperplane and a 2-D space hyperplane mapping; then they mapped this $p-2$ loops 1-D time hyperplane to a 1-D clock ticks. [1] [8] [9] [12] [13] [14] [15] mapped a p dimensional problem space to a 1-D time hyperplane and a $(p-1)$-D space hyperplane mapping.

A mapping is *correct* if it satisfies the following three conditions:

1. it preserves the data-dependence ordering of the original sequential algorithm,
2. the "right" data flows to the "right" place at the "right" time, and
3. there is no data collision in the data links of the systolic array.

It has been proved that a correct transformation of a p dimensional problem space to a 1-D time hyperplane and a $(p-1)$-D space hyperplane mapping satisfied conditions 1 and 2 [12]. In addition, we will show later in this report how to assure that it also satisfies condition 3. However, there have been no formal proofs that a specific transformation of a p dimensional problem space to a 1-D time hyperplane and an q-D space hyperplane mapping is correct, where $1 \leq q < p-1$, and in addition, there have been no formal proofs that a specific transformation of a p dimensional

problem space to a t loops 1-D time hyperplane and an s-D space hyperplane mapping is correct, where $p = t + s$ and $s \neq p - 1$. In fact, [2] [4] [10] [11] [16] did not show that their mappings had no data collisions in the data links. This is because the space-time mapping only requires a nonsingular mapping which preserves the data-dependence ordering of the original sequential algorithm. Thus, it only satisfies conditions 1 and 2 in the previous paragraph. Coincidently, there is no data collision in the data links when transforming a p dimensional problem space to a 1-D time hyperplane and a $(p-1)$-D space hyperplane mapping. However, when $p = 3$, the authors [6] gave examples showing that a nonsingular 1-D time hyperplane and a 1-D space hyperplane mapping as well as a nonsingular *2* loops 1-D time hyperplane and a 1-D space hyperplane mapping for a depth *3* loops matrix multiplication algorithm had data collisions in the data links. Therefore, this problem is nontrivial for the cases where $s \neq p - 1$.

Recently the authors gave necessary and sufficient conditions for transforming a p dimensional problem space to a 1-D time hyperplane and a 1-D space hyperplane mapping [6] [7]. There thus exist methods for transforming a p nested loop algorithm into a $(p-1)$-D systolic array and into a 1-D (linear) array. In this report, we close this gap by presenting necessary and sufficient conditions for transforming a p nested loop algorithm into a q-D systolic array, where $1 \leq q \leq p - 1$. We thus are able to synthesize q-D systolic array algorithms from nested loop algorithms in the general case, where $1 \leq q \leq p - 1$.

2 Mapping p Nested Loop Algorithms into q-D Systolic Arrays

2.1 Depth p Nested Loop Algorithms

For a model of a nested loop algorithm we use, see [6]. A p nested loop algorithm A_g comprised of three parts: (1) the loop index set, I^p, which is also called the problem space; (2) the set of variables, including input variables, output variables, and temporary variables; and (3) the sequence of statements in the loop body.

Then we considered the set of data-dependence vectors of the p nested loop algorithm. A data-dependence vector d_i of a variable can be viewed as difference of indices where a variable is used and where that variable was generated. Then, we further classified the data dependence vectors

according to Zero-One-Infinite property. For more details, see [6] [7].

For simplicity, in this report we assume that if $d_i = (d_{i1}, d_{i2}, \ldots, d_{ip})$, then $\gcd(d_{i1}, d_{i2}, \ldots, d_{ip}) = 1$. The more general case where $\gcd(d_{i1}, d_{i2}, \ldots, d_{ip}) \neq 1$ is beyond the scope of this report.

2.2 q-D Systolic Arrays

For brevity, in this report we only give a brief description of q-D systolic arrays. The topology structure of a q-D systolic array is similar to a q-D convex lattice. Each processor element (PE) in a q-D systolic array can be indexed by a q-tuple in q-D coordinate. In this structure, PEs are connected by data links. However, each PE can only be connected to its neighbors. Therefore, there are at most 3^q number of directions for data links. For example, in Fig. 1. there are at most $9 = 3^2$ directions in a 2-D array.

In order to implement fixed data stream velocity, each data link may or may not have a buffer, which consists of registers. The lengths of the buffers for different data links may be different. For convenience, we will use A_r to represent a q-D systolic array; K, the number of data links in each PE; t_i, the direction of data link i; and b_i, the size of the buffer corresponding to data link i. A typical 1-D array can be seen in Fig. 2.

2.3 Data Dependence Vectors, Data Links, and Data Streams

The authors in [6] [7] related data-dependence vectors to data links and data streams. In general, each data-dependence vector corresponds to a data link, which also corresponds to a data stream. Therefore, the number of data links must be at least equal to the number of data-dependence vectors. As we will associate a single dedicated data link with each data-dependence vector, for convenience, we will say that data-dependence vector d_i will correspond to data link i and data stream i.

2.4 Time Hyperplanes and Space Hyperplanes

It is our goal to assign each index of I^p to both a specific time instance and a specific q-D array location by means of a linear transformation. We can therefore describe the desired assignment as a linear mapping from p dimensions into $q + 1$ dimensions, where $p \geq q + 1 \geq 2$. Thus we map

the p-D space of indices into a $(q+1)$-D space: $(i_1, i_2, \ldots, i_p)^t \longmapsto (t, l_1, \ldots, l_q)$, where $(l_1, \ldots, l_q)$ state the location of the PE executing index $(i_1, i_2, \ldots, i_p)^t$ at time t.

Our algorithms, denoted by (H, S), can therefore be described by means of a mapping described by p nested loops on a $q+1$ dimensional space. H is a vector $(h_1, \ldots, h_p)$ and S is a matrix $\begin{pmatrix} s_{11} & \ldots & s_{1p} \\ \vdots & \ddots & \vdots \\ s_{q1} & \ldots & s_{qp} \end{pmatrix}$. Given an index $(i_1, i_2, \ldots, i_p)^t$, the mapping $\begin{pmatrix} \mathrm{H} \\ \mathrm{S} \end{pmatrix} (i_1, i_2, \ldots, i_p)^t$ is a $(q+1)$ dimensional vector:

$$\begin{pmatrix} h_1 & \ldots & h_p \\ s_{11} & \ldots & s_{1p} \\ \vdots & \ddots & \vdots \\ s_{q1} & \ldots & s_{qp} \end{pmatrix} (i_1, i_2, \ldots, i_p)^t = \begin{pmatrix} h_1 i_1 + \cdots + h_p i_p \\ s_{11} i_1 + \cdots + s_{1p} i_p \\ \vdots \quad + \ddots + \quad \vdots \\ s_{q1} i_1 + \cdots + s_{qp} i_p \end{pmatrix} = \begin{pmatrix} t \\ l_1 \\ \vdots \\ l_q \end{pmatrix}$$

where t and $(l_1, \ldots, l_q)$ specify the time and the q-D location of a data token with index $(i_1, i_2, \ldots, i_p)^t$. We refer to this mapping as a 1-D time hyperplane and a q-D space hyperplane q-D array algorithm (H,S). In general, the time hyperplane H and the space hyperplane S must satisfy certain constraints which we will state next.

3 Necessary and Sufficient Conditions

Let $1 \leq q \leq p-1$. A correct q-D array algorithm (H,S) that maps a p nested loop algorithm Ag into a q-D array Ar must preserve data-dependence relations, the right tokens must be in the right place at the right time, and in addition, data tokens must not collide in data links.

3.1 Necessary Conditions

We now consider conditions to be satisfied by (H,S) in order to assure a correct construction. There are four necessary conditions for (H,S).

1. H has to preserve the data-dependence relation, that is, if $I_2 - I_1 = d_i$ and $d_i \neq \vec{0}_p$ ($\vec{0}_p = (0,0,\ldots,0)^t$, a p-tuple zero vector) for all two indices I_1 and I_2, then I_2 must be executed after I_1. That is, $\mathrm{H}I_2 - \mathrm{H}I_1 > 0$, or, $\mathrm{H}(I_2 - I_1) = \mathrm{H}d_i > 0$. (In the following report, we also assume

$I_1 \neq I_2$ if we do not mention it.)

Condition 1 : $\mathbf{H}d_i > 0$ *must be true for all nonzero data-dependence vectors* d_i $(\neq \vec{0}_p)$.

Note: This follows immediately from [3] [5].

2. No two indices I_1 and I_2 can be mapped to the same PE at the same time, that is, $\mathbf{H}I_1 = \mathbf{H}I_2$ and $\mathbf{S}I_1 = \mathbf{S}I_2$ can not be both true at the same time.

Condition 2 : *If* I_1 *and* I_2 *are two indices of* I^p *then* $\begin{pmatrix} \mathbf{H} \\ \mathbf{S} \end{pmatrix} (I_2 - I_1) \neq \vec{0}_{q+1}$.

Note: (1) This condition is related to the nonsingularity condition of [4], [12], and others. (2) Generally, the space-time mapping only deals with Conditions 1 and 2.

3. We now consider the direction and the length of the delay buffer in each data link. Suppose a variable with the data-dependence vector d_i is generated in index $\bar{I}$ and will be used next time in index $\bar{I} + d_i$. Index $\bar{I}$ is executed in $PE_{\mathbf{S}\bar{I}}$ at time $\mathbf{H}\bar{I}$, and the index $\bar{I} + d_i$ will be executed in $PE_{\mathbf{S}(\bar{I}+d_i)}$ at time $\mathbf{H}(\bar{I} + d_i)$. Thus the corresponding token is in $PE_{\mathbf{S}\bar{I}}$ at time $\mathbf{H}\bar{I}$ and is in $PE_{\mathbf{S}(\bar{I}+d_i)}$ at time $\mathbf{H}(\bar{I} + d_i)$. We consider two cases:

1. $\mathbf{S}d_i \neq \vec{0}_q$.

   There must have a data link, say data link i of direction $t_i \neq \vec{0}_q$, connecting $PE_{\mathbf{S}\bar{I}}$ and $PE_{\mathbf{S}(\bar{I}+d_i)}$; in addition, $\mathbf{S}(\bar{I} + d_i) - \mathbf{S}\bar{I} = \mathbf{S}d_i = c_i t_i$ for some integer c_i. That is, this token flows through $PE_{\mathbf{S}\bar{I}}$, $PE_{\mathbf{S}\bar{I}+t_i}$, $PE_{\mathbf{S}\bar{I}+2t_i}$, $\cdots$, $PE_{\mathbf{S}\bar{I}+c_i t_i} = PE_{\mathbf{S}(\bar{I}+d_i)}$. As this token is delayed by a constant amount of time, say g, in each PE, $\mathbf{H}(\bar{I} + d_i) - \mathbf{H}\bar{I} = \mathbf{H}d_i = c_i g$.

   We now examine the data flow behavior of data stream i. Consider a token with data dependence vector d_i. From the discussion above, it follows that it will be delayed for g time while travelling through one PE. One time unit is allocated to the processing time and therefore we have to account for the remaining $g - 1$ time units. To accomplish that, we need

g shift registers in data link i; in addition, CPU in each PE only connects to the first shift register. Let b_i be the number of shift registers in data link i, we have $g = b_i$. Therefore, $\mathbf{H}d_i = \mathbf{c}_i b_i$.

2. $\mathbf{S}d_i = \vec{0}_q$.

   In this case, $PE_{\mathbf{S}I} = PE_{\mathbf{S}(I+d_i)}$. That is, the corresponding tokens with d_i are fixed in PEs (and thus tokens do not move in data link i). Therefore, $t_i = \vec{0}_q$.

   We now examine the tokens' behavior in data stream i. Since the tokens do not move in the data link, we need *local registers* instead of shift registers. Generally, if the tokens are input or output variables, then we need I/O ports for transferring them from/to the host computer. There are different approaches for solving this problem. We only mention two of the more interesting of such designs.

   If each PE has an additional I/O port for transferring tokens with the host computer, then we need only one local register for data link i. Because we can read (write) a token from (to) the host computer just before (after) it is used, generated, or regenerated. However, this design approach will incur $O(n^q)$ number of I/O ports, where n is the problem size.

   Another approach would be to preload all of these tokens before any execution and download all of these tokens after all executions. In this case we only need $O(n^{q-1})$ I/O ports, but the number of local registers in data link i must be at least the maximum number of different tokens used in data link i for each PE. In addition, because there may be more than one local register in data link i, we need memory addressing control hardware.

   Because the additional I/O ports are expensive, we sometimes choose designs with smaller number of I/O ports. Therefore, b_i, the number of local registers, is at least equal to the maximum number of different tokens used (generated) in data link i for each PE. Of course, we sometimes too avoid the space mapping S such that $\mathbf{S}d_i = \vec{0}_q$.

Condition 3 : *There are two cases:*

1. $\mathbf{S}d_i \neq \vec{0}_q$.

   Let b_i be the number of shift registers in data link i for each PE and t_i be the direction in data link i. Then, $\mathbf{S}d_i = c_i t_i$ and $\mathbf{H}d_i = c_i b_i$ for some integer c_i.

2. $\mathbf{S}d_i = \vec{0}_q$.

   Let b_i be the number of local registers in data link i for each PE and t_i be the direction in data link i. Then, $t_i = \vec{0}_q$ and b_i is the maximum number of different tokens used (generated) in data link i for each PE.

Note: (1) When $\mathbf{S}d_i \neq \vec{0}_q$, one also can used b_i local registers instead of b_i shift registers; however, in this case each PE needs additional memory addressing control hardware. (2) When $\mathbf{S}d_i = \vec{0}_q$, there are many approaches for determining b_i depending on whether there are additional I/O ports in each PE or not. There are, however, beyond the scope of this paper.

4. So far we have only examined the behavior of individual tokens. Now we will consider the possible interference between tokens of the same data stream. We will examine the case where $\mathbf{S}d_i \neq \vec{0}_q$ and $(I_2 - I_1) \neq md_i$ for all integers m. Then we will show that we cannot have $\mathbf{H}(I_2 - I_1)\mathbf{S}d_i = \mathbf{S}(I_2 - I_1)\mathbf{H}d_i$. as otherwise collisions would occur in data links. As this is rather non-intuitive, the authors in [6] gave an example of mapping a depth *3* loops matrix multiplication algorithm into 1-D (linear) arrays that neither $\mathbf{H_1} = (2,1,2)$ and $\mathbf{S_1} = (1,1,-2)$ nor $\mathbf{H_2} = \begin{pmatrix} 0 & 0 & 1 \\ 2 & 1 & 2 \end{pmatrix}$ and $\mathbf{S_2} = \begin{pmatrix} 1 & 1 & -2 \end{pmatrix}$ are feasible solutions, because data collisions occur in data links. However, as $(\mathbf{H_1}, \mathbf{S_1})$ and $(\mathbf{H_2}, \mathbf{S_2})$ satisfy Conditions 1 through 3, the space-time mapping by itself can not guarantee a correct mapping in the general case.

Condition 4 : *If $\mathbf{S}d_i \neq \vec{0}_q$ and $(I_2 - I_1) \neq md_i$ for all integers m, then $\mathbf{H}(I_2 - I_1)\mathbf{S}d_i \neq \mathbf{S}(I_2 - I_1)\mathbf{H}d_i$.*

Lemma 1 : *Condition 4 is necessary.*

Proof : The proof proceeds similarly to the proof of Lemma 8 in [6] replacing S which is a 1-D

hyperplane there by q-D hyperplane here. □

We summarize the preceding discussion in:

Theorem 2 : *Let* $1 \leq q \leq p-1$. *A q-D array algorithm* $(\mathbf{H},\mathbf{S})$ *that maps correctly a p nested loop algorithm Ag into a q-D array Ar must satisfy Conditions 1 through 4.* □

3.2 Sufficient Conditions

By assuming Conditions 1 through 4, we will show that $(\mathbf{H},\mathbf{S})$ will preserve the data-dependence relations, the right tokens will be in the right place at the right time, and in addition, no data tokens will collide in data links. Therefore, Conditions 1 through 4 are not only necessary, but also sufficient.

Theorem 3 : *Let* $1 \leq q \leq p-1$. *A q-D array algorithm* $(\mathbf{H},\mathbf{S})$ *from a p nested loop algorithm Ag that satisfies Conditions 1 through 4 maps Ag correctly into a q-D array Ar.*

Proof : Consider three cases, depending on the values of d_i and $\mathbf{S}d_i$.

1. $d_i = \vec{0}_p$.

   In this case, a token in data stream i is used or generated only once in some index $\bar{I} \in I^p$; therefore, it will not be used or generated in data stream i again. Thus, it has no dependence relation in data stream i. Next, as tokens are stored in local registers, CPUs can read them whenever they are needed. Third, since they are stored in local registers, they will not collide in data links.

2. $d_i \neq \vec{0}_p$ and $\mathbf{S}d_i = \vec{0}_q$.

   Unlike for case 1, although a token in data stream i is used or generated more than once in different indices in I^p, from Condition 1 $\mathbf{H}$ preserves the data-dependence ordering. Next, similar to case 1 tokens are stored in local registers; therefore, CPUs can read them whenever they are needed, and in addition, they also will not collide in data links.

3. $d_i \neq \vec{0}_p$ and $\mathbf{S}d_i \neq \vec{0}_q$.

   In this case, the proof proceeds similarly to the proof of Theorem 10 in [6] replacing S which is a 1-D hyperplane there by q-D hyperplane here. □

In general, Conditions 1 through 4 are independent of one another. However, when $q = p - 1$, Condition 4 can be derived from Condition 2.

Lemma 4 : *Let* $q = p - 1$, $\mathbf{S}d_i \neq \vec{0}_q$, *and* $(\mathbf{H}, \mathbf{S})$ *satisfy Condition 2. Then when* $I_2 - I_1 \neq md_i$ *for all integers* m, *the inequality* $\mathbf{H}(I_2 - I_1)\mathbf{S}d_i \neq \mathbf{S}(I_2 - I_1)\mathbf{H}d_i$ *is always true.*

Proof : We consider three cases:

1. $\mathbf{H}I_1 = \mathbf{H}I_2$.

   $\mathbf{H}(I_2 - I_1)\mathbf{S}d_i = (\mathbf{H}I_2 - \mathbf{H}I_1)\mathbf{S}d_i = \vec{0}_q$. From Condition 1, $\mathbf{H}d_i \neq 0$ and from Condition 2, $\mathbf{S}I_2 \neq \mathbf{S}I_1$. Therefore, $\mathbf{S}(I_2 - I_1)\mathbf{H}d_i = (\mathbf{S}I_2 - \mathbf{S}I_1)\mathbf{H}d_i \neq \vec{0}_q$.

2. $\mathbf{S}I_1 = \mathbf{S}I_2$.

   From Condition 2, it follows that $\mathbf{H}I_2 \neq \mathbf{H}I_1$. As we assume that $\mathbf{S}d_i \neq \vec{0}_q$, $\mathbf{H}(I_2 - I_1)\mathbf{S}d_i \neq \vec{0}_q$. However, as $\mathbf{S}I_2 = \mathbf{S}I_1$, $\mathbf{S}(I_2 - I_1)\mathbf{H}d_i = \vec{0}_q$.

3. $\mathbf{H}I_1 \neq \mathbf{H}I_2$ and $\mathbf{S}I_1 \neq \mathbf{S}I_2$.

   $\mathbf{H}(I_2 - I_1)$ and $\mathbf{H}d_i$ are two integers, $\mathbf{S}(I_2 - I_1)$ and $\mathbf{S}d_i$ are two $(p-1)$-tuple vectors. Assume by contradiction that $\mathbf{H}(I_2 - I_1)\mathbf{S}d_i = \mathbf{S}(I_2 - I_1)\mathbf{H}d_i$ for some $I_2 - I_1 \neq md_i$. Then $\mathbf{S}(I_2 - I_1)$ must be equal to $r\mathbf{S}d_i$ for some integer r and $\mathbf{H}(I_2 - I_1)$ must be equal to $r\mathbf{H}d_i$. (We use the assumption that $\gcd(d_{i_1}, d_{i_2}, \ldots, d_{i_p}) = 1$, where $d_i = (d_{i_1}, d_{i_2}, \ldots, d_{i_p})$.) We have $\mathbf{H}((I_2 - I_1) - rd_i) = 0$ and $\mathbf{S}((I_2 - I_1) - rd_i) = \vec{0}_{p-1}$. That is, $\begin{pmatrix} \mathbf{H} \\ \mathbf{S} \end{pmatrix}((I_2 - I_1) - rd_i) = \vec{0}_p$. However, from Condition 2, $\mathbf{H}$ and $\mathbf{S}$ are nonsingular, and in addition, $\mathbf{H}$ is of rank 1 and $\mathbf{S}$ is of rank $p - 1$ (because $q = p - 1$). Therefore, $\begin{pmatrix} \mathbf{H} \\ \mathbf{S} \end{pmatrix}$ is a basis. From the fact that a basis will not map a non-zero vector to a zero vector, $I_2 - I_1$ must be equal to rd_i. However, this contradicts our assumption that $I_2 - I_1 \neq md_i$ for all integers m. □

Theorem 5 : *Let $q = p - 1$. A q-D array algorithm* $(\mathbf{H}, \mathbf{S})$ *from a p nested loop algorithm Ag that satisfies Conditions 1 through 3 maps Ag correctly into a q-D array Ar.*

Proof : From Lemma 4, Condition 4 can be derived from Condition 2. Therefore, the rest of the proof is the same as that of Theorem 3, except that we do not need to consider data collisions in the data links. □

Theorem 5 shows that Conditions 1, 2, and 3 are both necessary and sufficient for transforming a p nested loop algorithm to a $(p-1)$-D systolic algorithm. Theorem 5 also explains why the space-time mapping and Condition 3 by themselves are sufficient for transforming a p nested loop algorithm to a $(p-1)$-D systolic algorithm.

References

[1] M. C. Chen, "The Generation of a Class of Multipliers: Synthesizing Highly Parallel Algorithms in VLSI," *IEEE Trans. Comput.*, vol. C-37, pp. 329-338, March 1988.

[2] C.-H. Huang, "The Mechanically Certified Derivation of Concurrency and Application to Systolic Design," Ph.D. dissertation, UT Austin, Austin, TX, 1987.

[3] R. M. Karp, R. E. Miller, and S. Winograd, "The Organization of Computations for Uniform Recurrence Equations," *JACM*, 14(3): pp. 563-590, July 1967.

[4] R. H. Kuhn, "Transforming Algorithms for Single-Stage and VLSI Architectures," in *Proc. of the workshop on Interconnection Networks for Parallel and Distributed Processing, IEEE CH1560-2,* pp. 11-19, 1980.

[5] L. Lamport, "The Parallel Execution of Do Loops," *CACM*, pp. 83-93, Feb. 1974.

[6] P.-Z. Lee and Z. M. Kedem, "Synthesizing Linear-Array Algorithms from Nested For Loop Algorithms," to appear in *IEEE Trans. Comput.*, preliminary version also available as NYU Computer Science TR-355, March 1988.

[7] P.-Z. Lee and Z. M. Kedem, "On High-Speed Computing with a Programmable Linear Array," to appear in *Supercomputing '88*, Kissimmee, FL, Nov. 1988, also available as NYU Computer Science TR-361, April 1988.

[8] G. Li and B. W. Wah, "The Design of Optimal Systolic Arrays," *IEEE Trans. Comput.*, vol. C-34, pp. 66-77, Jan. 1985.

[9] W. L. Miranker and A. Winkler, "Spacetime representations of computational structures," *Computing 32*, pp. 93-114, 1984.

[10] D. I. Moldovan, "On the Analysis of VLSI Systems," *IEEE Trans. Comput.*, vol. C-31, pp. 1121-1126, Nov. 1982.

[11] D. I. Moldovan, "On the design of algorithms for VLSI systolic arrays," in *Proc. IEEE*, vol. 71, pp. 113-120, Jan. 1983.

[12] D. I. Moldovan and J. A. Fortes, "Partitioning and Mapping Algorithms into Fixed Size Systolic Arrays," *IEEE Trans. Comput.*, vol. C-35, pp. 1-12, January 1986.

[13] P. Quinton, "Automatic synthesis of systolic arrays from uniform recurrent equations," in *Proc. 11th Annu. Symp. Comput. Architecture*, pp. 208-214, 1984.

[14] P. Quinton, "Mapping Recurrences on Parallel Architectures," in *Third Int. Conf. on Supercomputing*, Boston, MA, May 15-20, 1988.

[15] S. K. Rao, "Regular Iterative Algorithms and Their Implementations on Processor Arrays," Ph.D. dissertation, Stanford Univ., Stanford, CA, 1985.

[16] Y. Wong and J. Delosme, "Optimal Systolic Implementations of N-Dimensional Recurrences," in *IEEE Proc. ICCD*, pp. 618-621, 1985

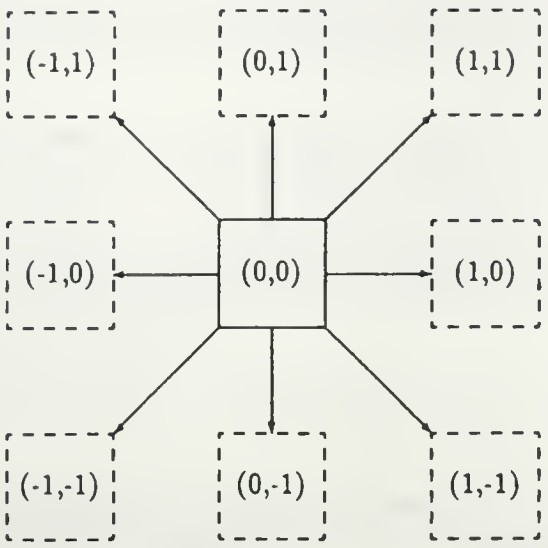


Fig. 1. There are at most nine directions for data links in a 2-D array. Eight of the nine connect to the neighbor PEs. One of the nine "is fixed in PEs."

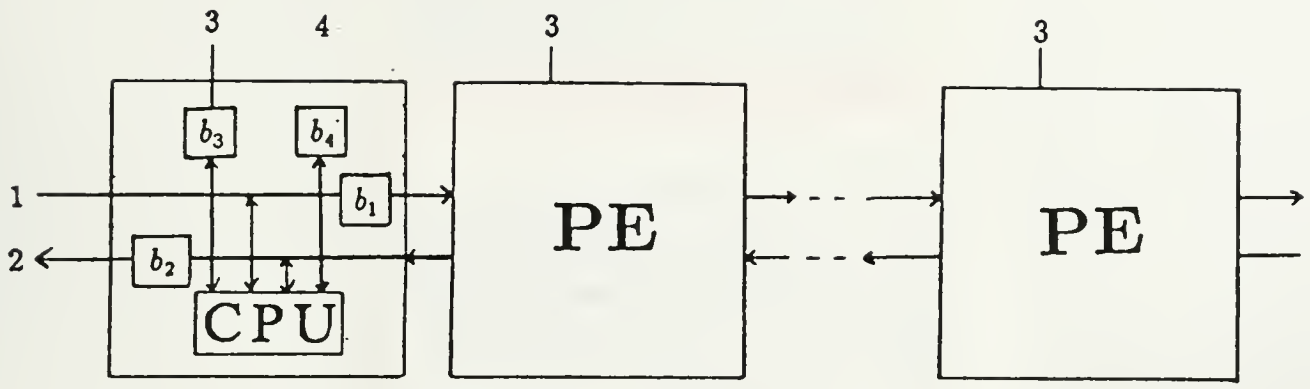


Fig. 2. A linear array comprised of PEs. There are four types of data links, which have three directions. A data link of type 1 "flows" from left to right, a data link of type 2 "flows" from right to left, data links of types 3 or 4 are "fixed" in PEs; the former need an I/O port and the latter do not. b_1, b_2, b_3, and b_4 are lengths of the buffers.

NYU COMPSCI TR-395
Lee, PeiZong
Mapping nested loop
algorithms into multi-
dimensional... c.2

DATE DUE	BORROWER'S NAME